OKTICKET S.L.

API - Guía de uso

v1.2 - 28 de Agosto de 2024

OBJETIVO Y ALCANCE

El presente documento pretende servir de guía inicial de uso de la API de Okticket. Está especialmente orientada para usuarios integradores que pretendan hacer uso de la API de Okticket.

El documento cubre ejemplos básicos de uso, autenticación, consulta, creación, modificación y borrado de elementos.

Se complementa con una colección de la aplicación POSTMAN, disponible en https://documenter.getpostman.com/view/5962582/RzffK9uS, donde figuran todas las llamadas disponibles, todas ellas operativas dentro de un entorno de desarrollo, también compartido en dicha colección.

VISIÓN GENERAL DE LA API

Okticket expone una API de tipo REST con comunicaciones en formato JSON, y autenticación Oauth2.

AUTENTICACIÓN

Previamente a cualquier llamada a la API, hay que obtener un **token**, que servirá para *autorizar* el resto de llamadas.

Obtención del primer token

Con los datos de autenticación facilitados por Okticket, hay que realizar una llamada POST a la URL:

https://**HOST**/**CONTEXT**/oauth/token

Donde:

- HOST: dirección del host donde se encuentra la API.
    - En el entorno de desarrollo es: *dev.okticket.es*
- CONTEXTO: contexto de publicación de la API. Puede variar entre entornos
    - En el entorno de desarrollo es: *api/public*

Y con los siguientes parámetro en el cuerpo de la llamada (por ejemplo en formato *multipart/form-data*)

- grant_type = password

- client_id = <**id de cliente, proporcionado por Okticket**>
- client_secret = <**secreto de cliente, proporcionado por Okticket**>
- username = <**usuario, proporcionado por Okticket**>
- password = <**contraseña, proporcionada por Okticket**>
- scope=*

Si la autenticación tiene éxito, la respuesta tendrá estado 200 - OK y el cuerpo tendrá un objeto JSON con un contenido similar al siguiente:

```
{
    "token_type": "Bearer",
    "expires_in": 1800,
    "access_token": "eyJ0eXAiOiJKc71335dcbf9caac3c1bdc6be105c78e7...f-CY",
    "refresh_token": "def50200d20be599ced63caf79...a5ea50d8a23b10fc63fe2"
}
```

Donde:

- token_type: Siempres será de tipo Bearer
- expires_in: Tiempo de vigencia del token, en segundos.
- **access_token**: Token de acceso. Será el token a usar en el **resto de llamadas a la API**.
- **refresh_token**: Token de refresco. Token para obtener nuevos tokens sin necesidad de volver a autenticarse.

Si la autenticación falla, la respuesta puede ser:

Código 401 - Unauthorized

Alguno de los datos proporcionados no sirve para autenticar el acceso.

Código 400 - Bad request

La petición no tiene el formato esperado (faltan campos o tienen un valor no esperado)

El token tiene un tiempo de vida de 30 minutos, aunque puede variar. Pasados esos 30 minutos, hay que solicitar de nuevo el token, bien invocando de nuevo al método descrito, o bien al método para obtener un nuevo token de refresco. El token de refresco tiene un tiempo de validez considerablemente mayor que el token de acceso (del orden de varios días).

Uso del token de acceso

Una vez obtenido el token de acceso, **todas** las llamadas a la API de Okticket deberán incluir **obligatoriamente** la cabecera:

Authorization: Bearer <**token de acceso obtenido anteriormente**>

Si no se incluye dicha cabecera o el token empleado no es válido, o está caducado, se obtendrá una respuesta con código 401 - Unauthorized.

Uso del token de refresco

Para obtener un nuevo token sin tener que proporcionar usuario y password de nuevo, se puede emplear el token de refresco obtenido antes.

Para ello hay que hacer una llamada al mismo endpoint anterior, /oauth/token, con los siguientes datos:

- grant_type = refresh_token
- client_id = <**id de cliente, proporcionado por Okticket**>
- client_secret = <**secreto de cliente, proporcionado por Okticket**>
- refresh_token= <**refresh_token, obtenido anteriormente**>
- scope=*

Si la autenticación tiene éxito, la respuesta tendrá estado 200 - OK y el cuerpo tendrá un objeto JSON con un contenido similar al siguiente:

```
{
    "token_type": "Bearer",
    "expires_in": 1800,
    "access_token": "eyJ0eXAiOiJKc71335dcbf9caac3c1bdc6be105c78e7...f-CY",
    "refresh_token": "def50200d20be599ced63caf79...a5ea50d8a23b10fc63fe2"
}
```

Es decir, se habrá generado un nuevo par TOKEN DE ACCESO, TOKEN DE REFRESCO que habrá que usar a partir de este momento. Los tokens anteriores quedarán **revocados** a partir de este instante y no podrán volver a usarse.

Errores

Si se intenta obtener un token de ACCESO a partir de un token de REFRESCO que ya ha sido usado y por tanto está revocado:

```
{
    "error": "invalid_request",
    "error_description": "The refresh token is invalid.",
    "hint": "Cannot decrypt the refresh token",
    "message": "The refresh token is invalid."
}
```

FORMATO DE LLAMADAS

De modo general, todas las URLs expuestas por a la API siguen el siguiente formato:

METHOD https://**HOST**/**CONTEXT**/api/**ENTIDAD**/[**ID**[/**ACTION**]]

Donde:

- METHOD, puede ser uno de los siguientes
    - GET: Generalmente usado para consultar información
    - POST: Para crear nuevos elementos
    - PATCH / PUT: Para actualizar o reemplazar elementos
    - DELETE: Para eliminar elementos
- HOST: dirección del host donde se encuentra la API.
    - En el entorno de desarrollo es: *dev.okticket.es*
- CONTEXTO: contexto de publicación de la API. Puede variar entre entornos
    - En el entorno de desarrollo es: *api/public*
- ENTIDAD: nombre de la entidad sobre la que actuar. Puede ser, por ejemplo:
    - users (usuarios)
    - companies (empresas)
    - expenses (gastos)
    - reports (hojas de gasto)
- ID (opcional): identificador único del recurso sobre el que actuar. Solo se incluirá cuando se necesite actuar sobre un recurso en concreto. Suelen ser numéricos (1, 34, 5810) o alfanuméricos (5a5f097bd7fe0131fb3c7872)
- ACTION (opcional): De uso poco frecuente, sirve para distinguir qué acción se quiere realizar sobre un recurso concreto, por tanto, requiere de ID.

CUERPO DEL MENSAJE

Se puede pasar en formato JSON o como X-WWW-FORM-URLENCODED, codificado en la URL. Se recomienda la primera opción, por simplicidad, pero ambas se admiten.

Ejemplos, ambos válidos (ojo, se omite la cabecera de autorización):

1. Actualizar los datos de un usuario

   PATCH http://dev.okticket.es/api/users/4

   Content-Type: application/json

```
{
      "name" : "Remedios López",
      "ids_role": 3,
      "ids_companies" : {
            "1" : {
                  "id_role" : 2
            }
      }
}
```

2. Crear un usuario

   POST http://dev.okticket.es/api/users

   Content-Type: application/x-www-form-urlencoded

```
name=Remedios%20Lopez\
&email=remedios%40test.com\
&password=supersecret\
&id_role=3\
&ids_companies%5B1%5D%5Bid_role%5D=3
```

RESPUESTA

La respuesta siempre contendrá un mensaje en formato JSON (Content-Type: application/json), y su formato variará en función del tipo y resultado de la operación realizada.

Consultas (GET)

200 - OK:

- data: Objeto o array de objetos consultados
- status: ok
- meta: Objeto con metainformación de la consulta, como página actual, número total de resultados, etc.

```
{
    "data": [
        {
            "id": 1,
            "name": "Básico",
            "company_id": null
        },
        {
```

```
            "id": 2,
            "name": "Calidad",
            "company_id": 4937
        }
    ],
    "meta": {
        "current_page": 1,
        "from": 1,
        "last_page": 2,
        "per_page": 2,
        "to": 2,
        "total": 4,
        "search": {}
    },
    "status": "ok"
}
```


Creación (POST)

Código 201 - Created:

- data: El objeto creado
- status: ok


```
{
    "data": {
        "name": "Santiago",
        "email": "test19@test.com",
        "id_role": "3",
        "updated_at": "2019-04-22 14:34:32",
        "created_at": "2019-04-22 14:34:32",
        "id": 17400
    },
    "status": "ok"
}
```


Modificación (PATCH/PUT)

Código 200 - OK

- data: El objeto modificado
- status: ok


```
{
    "data": {
        "id": 1,
        "name": "Santiago",
        "email": "santiago@sysmbiosis.com",
        "id_role": 1,
        "status_id": 2,
        "company_type_id": 1,
        "currency": 1,
```

```
        "distance_unit_price": "0.19",
        "app_version": "1.0.21",
        "legal_texts_version": "1",
        "pop_checked": 1,
        "tyc_checked": 1,
        "created_at": "2017-08-10 12:00:00",
        "updated_at": "2018-12-18 14:53:44",
        "deleted_at": null,
        "active_company": null,
        "active_department": null,
        "custom_id": null,
        "language": null
    },
    "status": "ok"
}
```


Borrado (DELETE)

Código 204 - No content

En este caso no se devuelve resultado, si el borrado se hizo correctamente.

Errores

Código 422 - Unprocessable entity

La entidad sobre la que se pretende actuar (consultar, crear, modificar, etc) no ha podido ser procesada correctamente. En este caso se devolverá un mensaje descriptivo:


```
{
    "message": "The given data was invalid.",
    "errors": {
        "email": [
            "El valor ya está en uso."
        ]
    }
}
```


Otros errores

Código 401 - Unauthorized

> Autorización no proporcionada o no válida.

Código 403 - Forbidden

> Se ha intentado intenta realizar una operación no permitido por algún motivo (permisos, estado del recurso, etc)

Código 404 - Not found

El recurso solicitado no existe o no se ha encontrado.

Código 405 - Method Not Allowed

El método especificado no está permitido. Cuando se intenta hacer un POST sobre un recurso que solo admite GET, por ejemplo.

Otras consideraciones importantes

Cabecera COMPANY

Junto con los datos de acceso a la API de Okticket, se proporcionará el acceso a una o varias empresas con datos en Okticket. Cada una se identificará mediante un código numérico único.

Todas las llamadas a la API deben ir acompañadas de la empresa contra la que se esté haciendo la consulta, creación o modificación, para evitar posibles errores o datos mezclados, aunque solo se tenga acceso a una única. Si no se proporciona, un resultado puede ser que las acciones que se intenten realizar no estén autorizadas.

Para ello hay que proporcionar el código de la empresa en la cabecera COMPANY:

company: <**codigo_empresa**>

Diferencias entre PATCH / PUT

Como norma general, se usará el método PATCH para actualizar los recursos mediante la API, ya que es el método ideal para modificar solo aquellos campos que se quieren modificar.

El método PUT reemplaza todo los campos, de modo que aquellos que no se incorporen en la llamada se borrarán, con la consiguiente pérdida de información si no se realiza la llamada correctamente.

Paginación

Por defecto, los métodos GET susceptibles de devolver una lista de objetos en su campo `data`, devolverá los datos paginados, devolviendo solo la primera página de resultados disponibles.

Recordemos el campo meta que se obtiene en las respuestas de los métodos GET de la API:

```
"meta": {
      current_page: 2
      from: 51
      last_page: 4854
      path: "https://api.okticket.es/api/expenses"
```

```
        per_page: 50
        to: 100
        total: 242688
    }
```

Este campo proporciona la siguiente información:

- *current_page*: página actual
- *last_page*: última página disponible
- *from*: índice del primer elemento devuelto en la página actual
- *to*: índice del último elemento devuelto en la página actual
- *per_page*: número de elementos por página
- *total*: número total de elementos disponibles

Para obtener todas las páginas, es decir, todos los elementos de una vez, basta con añadir a la URL el parámetro **paginate=false.** Por ejemplo:

```
https://api.okticket.es/api/expenses?paginate=false
```

Si lo que se desea es obtener los datos de otra página, basta con indicar en la URL el número de página deseada, con **page=N.** Por ejemplo:

```
https://api.okticket.es/api/expenses?page=2
```

Por último, si lo que se desea es obtener más resultados por página, hay que indicar el número máximo de resultados por página, con el parámetro **limit=M**. Por ejemplo:

```
https://api.okticket.es/v2/public/api/expenses?page=2&limit=30
```

Límite de llamadas por minuto

Otro dato importante a tener en cuenta es que la API de Okticket cuenta con un límite máximo de llamadas por minuto por cliente, para evitar saturaciones, accidentales o no, del servicio. Dicho límite varía por entorno, y puede comprobarse en las cabeceras de todas las respuestas que proporciona la API, en concreto en las cabeceras:

- *X-RateLimit-Limit*: Indica el límite máximo de llamadas por minuto, para el cliente actual.
- *X-RateLimit-Remaining*: Indica el límite de llamadas que quedan disponibles en el minuto actual.

Si el dato proporcionado por *X-RateLimit-Remaining* llega a cero, la API empieza a devolver errores por límite de llamadas superado (*Too Many Requests*). Basta con esperar al siguiente minuto. El límite se resetea con dicha frecuencia.

Entidades disponibles

Entidades

Se listan a continuación las principales entidades disponibles en Okticket y accesibles vía API. Se recuerda que en la colección de Postman proporcionada, hay múltiples ejemplos de llamadas para cada entidad, con los diversos métodos disponibles.

Empresas

La unidad central que engloba al resto de elementos. Con el acceso a la API se proporcionará el acceso a una o varias empresas, en función de las necesidades y configuración del cliente o clientes que vayan a hacer uso de la API.

Una empresa tiene varios usuarios, y todas las hojas de gastos, gastos, centros de coste, etc. estarán siempre asociados a una empresa.

Se recuerda una vez más que el código de la empresa con la que se esté trabajando debe pasarse siempre en la cabecera `company` de todas las llamadas a la API.

Usuarios

Cada empresa tiene N usuarios, y cada usuario tiene asociado un rol concreto dentro de la empresa. Los roles por defecto son 2:

- 3 - Empleado. Rol por defecto, solo tiene permiso para crear y manejar sus propios gasto y hojas de gastos
- 2 - Administrador de empresa. Tiene permisos adicionales para crear usuarios, y consultar los datos de todos los usuarios de la empresa, por ejemplo.

Existen otros roles, menos frecuentes, pero que pueden llegar a necesitar:

- 6 - Validador o jefe de equipo. Rol asignado a usuarios que validan los gastos de un grupo de usuarios.
- 5 - Usuario inactivo. Rol asignado a aquellos usuarios que ya no tienen acceso a la aplicación, pero que sus datos siguen siendo necesarios para la empresa.

Gastos

Un usuario imputa gastos mediante la aplicación y puede consultarlos mediante la propia aplicación o mediante el gestor web. Un gasto puede ser de tres tipos:

- 0: Ticket

- 1: Factura
- 2: Kilometraje

Además, tiene asignada una categoría. Las que existen por defecto son:

- 1 - Restauración
- 2 - Aparcamiento
- 3 - Peaje
- 4 - Transporte
- 5 - Alojamiento
- 6 - Gasolina
- 0 - Otros

Un gasto puede estar asociado o no a una hoja de gastos.

Hojas de gasto

Un usuario puede agrupar sus gastos en hojas de gasto que él mismo puede crear (desde la App o desde el gestor web).

Centros de coste

Opcionalmente, un gasto puede estar asociado a un centro de coste.

Los centros de coste se crean por empresa, Okticket no dispone de ninguno por defecto. Un centro de coste tiene los siguientes campos:

- descripción
- código (opcional)
- activo: por defecto, todos los centro de coste se crean activos
- global: indica si el centro de coste es visible para todos lo usuarios de la empresa o por el contrario, si hay que asignarlo únicamente a aquellos usuarios que vayan a usarlo.

Categorías de gastos

Una empresa puede crear sus propias categorías de gastos, y agruparlas, anidándolas, hasta un máximo de 2 niveles. Para indicar que una categoría es hija de otra, se usa el campo parent, que contendrá el código de la categoría padre. Si una categoría es padre, no pueden crearse gastos de dicha categoría: hay que asignar una categoría hija a cambio.

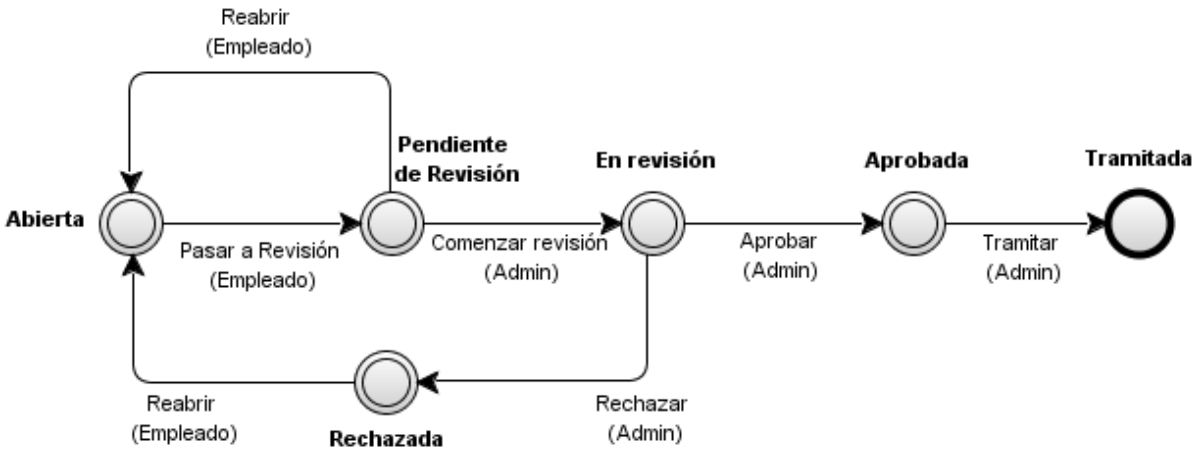
Estados de hoja de gastos

Las hojas de gasto son un elemento especial, pues tienen estados y por lo tanto tienen un ciclo de vida propio que pasa por esos estados. Los estados comunes son:

- Abierta (ID 0):
    - Estado inicial de todas las hojas de gastos
- Pendiente de revisión (ID 1)
    - El propietario de la hoja ha indicado que la hoja está lista para revisión, pero el validador aún no ha comenzado la revisión.
- En revisión (ID 2)
    - El validador ha comenzado la revisión. El propietario de la hoja no puede hacer cambios sobre la misma.
- Aprobada (ID 5)
    - El validador ha aprobado la hoja.
- Rechazada (ID 3)
    - El validador ha rechazado la hoja.
- Tramitada (ID 4)
    - El usuario tramitador ha tramitado la hoja. Es el estado final de la hoja, donde ya no admite más cambios por ningún usuario.

El ciclo de vida típico de una hoja de gastos es el siguiente: un usuario imputa todos los gastos que estime necesarios en una hoja de gastos en estado *Abierta*; una vez concluya puede pasar la hoja al estado *Pendiente de revisión*. La persona encargada de validar sus gastos podrá pasarla a su vez a *En revisión* y una vez acabada dicha revisión podrá pasarla al estado de *Aprobada*. Una vez aprobada, el encargado de tramitar las hojas de gastos de la empresa (puede ser el mismo usuario validador u otro distinto) podrá pasarla a su estado final *Tramitada.* En el caso de que el validador marque la hoja como *Rechazada*, el propietario de la misma, tendrá que reabrirla y repetir el proceso.

A modo de resumen se muestra a continuación el flujo de cambios de estados posibles.



Este es el flujo por defecto de Okticket, si bien puede cambiarse para adaptarse a las necesidades de validación de cada empresa, pudiendo incluso existir varios flujos de validación distintos dentro una misma empresa.